More Reliable Protein NMR Peak Assignment via Improved 2-Interval Scheduling (New Aspects of Theoretical Computer Science)

AUTHOR(S):

Chen, Zhi-Zhong; Jiang, Tao; Lin, Guohui; Rizzi, Romeo; Wen, Jianjun; Xu, Dong; Xu, Ying

CITATION:

Chen, Zhi-Zhong ...[et al]. More Reliable Protein NMR Peak Assignment via Improved 2-Interval Scheduling (New Aspects of Theoretical Computer Science). 数理解析研究所講究録 2003, 1325: 175-180

ISSUE DATE:

2003-05

URL:

http://hdl.handle.net/2433/43193

数理解析研究所講究録 1325 巻 2003 年 175-180

More Reliable Protein NMR Peak Assignment via Improved 2-Interval Scheduling

Zhi-Zhong Chen * Tao Jiang † Guohui Lin ‡ Romeo Rizzi § Jianjun Wen ¶
Dong Xu ‖ Ying Xu **

Abstract

Protein NMR peak assignment refers to the process of assigning a group of "spin systems" obtained experimentally to a protein sequence of amino acids. The automation of this process is still an unsolved and challenging problem in NMR protein structure determination. Recently, protein NMR peak assignment has been formulated as an *interval scheduling* problem, where a protein sequence $\mathcal{P}$ of amino acids is viewed as a discrete time interval $\mathcal{I}$ (the amino acids on $\mathcal{P}$ one-to-one correspond to the time units of $\mathcal{I}$), each subset S of spin systems that are known to originate from consecutive amino acids from $\mathcal{P}$ is viewed as a "job" j_S, the preference of assigning S to a subsequence P of consecutive amino acids on $\mathcal{P}$ is viewed as the profit of executing job j_S in the subinterval of $\mathcal{I}$ corresponding to P, and the goal is to maximize the total profit of executing the jobs (on a single machine) during $\mathcal{I}$. The interval scheduling problem is Max SNP-hard in general; but in the real practice of protein NMR peak assignment, each job j_S usually requires at most 10 consecutive time units, and typically the jobs that require one or two consecutive time units are the most difficult to assign/schedule. In order to solve these most difficult assignments, we present an efficient $\frac{13}{7}$-approximation algorithm for the special case of the interval scheduling problem where each job takes one or two consecutive time units. Combining this algorithm with a greedy filtering strategy for handling long jobs (*i.e.* jobs that need more than two consecutive time units), we obtain a new efficient heuristic for protein NMR peak assignment. Our experimental study shows that the new heuristic produces the best peak assignment in most of the cases, compared with the NMR peak assignment algorithms in the recent literature. The above algorithm is also the first approximation algorithm for a nontrivial case of the classical (weighted) interval scheduling problem that breaks the ratio 2 barrier.

1 Introduction

Due to the efforts of structural genomics [8], the NMR (nuclear magnetic resonance) technique has been used as a high-throughput technology to solve protein structures at a genome scale. Typically, protein structure determination via NMR involves the following steps:

- NMR spectral data generation, which produces
  - resonance peaks corresponding to amino acids in the target protein sequence. Peaks corresponding to a common amino acid are grouped into a *spin system*;
  - certain geometric relationships (*e.g.* distances and angles) between the spin systems;
- Peak picking, which identifies "real" resonance peaks (peaks generated from protein atoms rather than noise) from NMR spectral maps.
- Peak assignment, which assigns resonance peaks, typically peak groups, to individual residues of the target protein sequence.
- Structural restraint extraction, which extracts inter-residue distances, dihedral angles, etc., based on the peak assignment.
- Structure calculation, which calculates the protein structure, using molecular simulation and energy minimization, under the identified NMR restraints.

Among the five steps, the third one (namely, NMR peak assignment) is very time consuming. The process usually takes weeks or sometimes even months of manual work in order to produce a nearly complete assignment. The automation of the assignment process is still an unsolved and challenging problem in NMR protein structure determination.

Two key pieces of information form the foundation of NMR peak assignment:

*Department of Mathematical Sciences, Tokyo Denki University, Hatoyama, Saitama 350-0394, Japan. Part of work done while visiting at University of Alberta. Email: chen@r.dendai.ac.jp.

†Department of Compute Science, University of California, Riverside, CA 92521, and Shanghai Center for Bioinformation Technology. Email: jiang@cs.ucr.edu.

‡Department of Computing Science, University of Alberta, Edmonton, Alberta T6G 2E8, Canada. Email: ghlin@cs.ualberta.ca.

§Dipartimento di Informatica e Telecomunicazioni, Università di Trento, Italy Email: romeo@science.unitn.it

¶Department of Computer Science, University of California, Riverside, CA 92521. Email: wjianju@cs.ucr.edu.

‖Protein Informatics Group, Life Sciences Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6480, USA. Email: xud@ornl.gov.

**Protein Informatics Group, Life Sciences Division and Computer Sciences and Mathematics Division, Oak Ridge National Laboratory. Oak Ridge, TN 37831-6480, USA. Email: xyn@ornl.gov.

- The likelihood (or weight) of the matching between a spin system and an amino acid on the protein sequence.
- The sequential adjacency (*i.e.*, consecutivity) information of some subsets of spin systems (*i.e.*, each such subset of spin systems should correspond to a subsequence of consecutive amino acids on the host protein sequence). Each maximal such subset is called a *segment of spin systems*. It is worth noting that each segment usually consists of at most 10 spin systems.

In a recently developed computational framework [10], the NMR peak assignment problem has been formulated as a (weighted) *interval scheduling* problem[1] as follows. A protein sequence $\mathcal{P}$ of amino acids is viewed as a discrete time interval $\mathcal{I}$ (the amino acids on $\mathcal{P}$ one-to-one correspond to the time units of $\mathcal{I}$). Each segment S of spin systems is viewed as a job j_S. Each job j_S requires $|S|$ consecutive time units of $\mathcal{I}$ (this corresponds to the requirement that the spin systems in S should be assigned to $|S|$ consecutive amino acids on $\mathcal{P}$). For each time unit t of $\mathcal{I}$, the profit $w(j_S, t)$ of starting job j_S at time unit t and finishing at time unit $t + |S| - 1$ of $\mathcal{I}$ corresponds to the preference of assigning the spin systems in S to those $|S|$ consecutive amino acids on $\mathcal{P}$ that correspond to the time units $t, t+1, \ldots, t+|S|-1$. Given $\mathcal{I}$, the jobs j_S, and the profits $w(j_S, t)$, our goal is to maximize the total profit of the executed jobs (*i.e.* we want to find a maximum-likelihood assignment of the given spin systems to the amino acids on $\mathcal{P}$).

Unfortunately, the interval scheduling problem is Max SNP-hard [3, 4]. Indeed, for every integer $k \geq 2$, the special case of the interval scheduling problem (called the *k-interval scheduling problem* or *k-ISP* for short) where each job requires at most k consecutive time units is Max SNP-hard. On the other hand, several 2-approximation algorithms for the interval scheduling problem are known [2, 1, 3, 4]. Although these algorithms are theoretically sound, applying them to protein NMR peak assignment produces unsatisfactory assignments as demonstrated in [3]. A major reason why these algorithms do not have good performance in protein NMR peak assignment is that they ignore the following important observation:

- In the real practice of protein NMR peak assignment, long segments S of spin systems are typically easier to assign than shorter segments. In fact, many long segments have unique matches. On the other hand, segments consisting of one or two spin systems are often very difficult to assign.

The above observation suggests the following heuristic framework for protein NMR peak assignment: first try to assign segments consisting of at least $k+1$ spin systems for some small integer k (say, $k = 2$), and then solve an instance of k-ISP. In [7], we have presented such a heuristic and have shown that it is very effective for protein NMR peak assignment. A major drawback of the heuristic in [7] is that it uses an inefficient branch-and-bound algorithm for k-ISP.

In order to improve the efficiency of the heuristic in [7], we present a new approximation algorithm for 2-ISP in this paper. This algorithm achieves an approximation ratio of $\frac{13}{7}$ and is the first approximation algorithm for a nontrivial case of the classical interval scheduling problem that breaks the ratio 2 barrier.[2] Our algorithm is combinatorial and quite nontrivial – it consists of four separate algorithms and outputs the best solution returned by them. The main tool used in the algorithm design is maximum-weight bipartite matching and careful manipulation of the input instance. Since the algorithm is combinatorial, it is easy to implement and runs very fast in practice. Substituting the new algorithm for the branch-and-bound algorithm in the heuristic in [7], we obtain a new heuristic for protein NMR peak assignment.[3] We have performed extensive experiments on 70 instances of (pseudo) real NMR data derived from 14 proteins to evaluate the performance of our new heuristic in terms of (i) the weight of the assignment and (ii) the number of correctly assigned resonance peaks. The experimental results show that not only does the new heuristic run very fast, it also produces the best peak assignment on most of the instances, compared with the protein NMR peak assignment algorithms in the recent literature [3, 4, 7, 10].

2 A new approximation algorithm for 2-ISP

Let $\mathcal{I}$ be the given discrete time interval. Without loss of generality, we may assume that $\mathcal{I} = [0, I]$. Let $\mathcal{J}_1 = \{v_1, v_2, \ldots, v_{n_1}\}$ be the given set of jobs requiring one time unit of $\mathcal{I}$. Let $\mathcal{J}_2 = \{v_{n_1+1}, v_{n_1+3}, \cdots, v_{n_1+2n_2-1}\}$ be the given set of jobs requiring two contiguous time units of $\mathcal{I}$. Note that $n_1 + n_2$ is the total number of given jobs. For each $1 \leq i \leq I$, let u_i denote the time unit $[i-1, i]$ of $\mathcal{I}$. Let $U = \{u_i \mid 1 \leq i \leq I\}$. Let $\mathcal{J}_2' = \{v_{n_1+2}, v_{n_1+4}, \ldots, v_{n_1+2n_2}\}$. Let $V = \mathcal{J}_1 \cup \mathcal{J}_2 \cup \mathcal{J}_2'$. We construct an edge-weighted bipartite graph G with color classes U and V as follows: For every $v_j \in \mathcal{J}_1$ and every $u_i \in U$ such that the profit of executing job v_j in time unit u_i is positive, (u_i, v_j) is an edge of G and its weight is the profit. Similarly, for every $v_j \in \mathcal{J}_2$ and every $u_i \in U$ such that the profit of executing job v_j in the two time units u_i, u_{i+1} is positive, both (u_i, v_j) and (u_{i+1}, v_{j+1}) are edges of G and the weight of each of them is half the profit.

A *constrained matching* of G is a matching M of G such that for every $u_i \in U$ and every $v_j \in \mathcal{J}_2$, $(u_i, v_j) \in M$ if and only if $(u_{i+1}, v_{j+1}) \in M$. The objective of 2-ISP is equivalent to finding a maximum-weight constrained matching in G. For each edge (u_i, v_j) of G, let $w(u_i, v_j)$ denote the weight of the edge. For convenience, let $w(u_i, v_j) = 0$ for all $(u_i, v_j) \notin E$. For a (constrained or unconstrained) matching M of G, let $w_1(M)$ (respectively, $w_2(M)$) denote the total weight of edges $(u_i, v_j) \in M$ with $v_j \in \mathcal{J}_1$ (respectively, $v_j \in \mathcal{J}_2 \cup \mathcal{J}_2'$); let $w(M) = w_1(M) + w_2(M)$.

[1] In [10] it was called the *constrained bipartite matching* problem.

[2] For *unweighted* ISP where the profit of executing a job at each specific time interval is either 0 or 1 (independent of the job's length), Chuzhoy et al. [5] gave a 1.582-approximation algorithm. In this paper, our interest is in the weighted problem.

[3] The program is available to the public upon request to the authors.

Let M^* be a maximum-weight constrained matching in G. In Sections 2.1, 2.3 through 2.5, we will design four algorithms each outputting a constrained matching in G. The algorithm in Section 2.5 is the main algorithm and is quite sophisticated. We will try to find a large constant ϵ such that the heaviest one among the four output matchings is of weight at least $(\frac{1}{2}+\epsilon)w(M^*)$. It will turn out that $\epsilon = \frac{1}{26}$. So, fix $\epsilon = \frac{1}{26}$ for the discussions in the rest of this section.

2.1 Algorithm 1

This algorithm will output a constrained matching of large weight when $w_2(M^*)$ is relatively large compared with $w_1(M^*)$. We first explain the idea behind the algorithm. Suppose that we partition the time interval $\mathcal{I}$ into shorter intervals, called *basic intervals*, in such a way that each basic interval, except possibly the first and the last (which may possibly consist of 1 or 2 time units), consists of 3 time units. There are exactly three such partitions of $\mathcal{I}$. Denote them by P_0, P_1, and P_2, respectively. With respect to each P_h with $0 \le h \le 2$, consider the problem $\mathcal{Q}_h$ of finding a constrained scheduling which maximizes the total profit of the executed jobs, but subject to the constraint that each basic interval in P_h can be assigned to at most one job and each executed job should be completed within a single basic interval in P_h. It is not so hard to see that each problem $\mathcal{Q}_h$ requires the computation of a maximum-weight (unconstrained) matching in a suitably constructed bipartite graph, and hence can be solved in polynomial time.

We claim that among the three problems $\mathcal{Q}_h$, the best one gives a scheduling by which the executed jobs achieve at least a total profit of $\frac{1}{3}w_1(M^*)+\frac{2}{3}w_2(M^*)$. This claim is actually easier to see, if we refer to a more constrained scheduling problem $\mathcal{Q}'_h$ than $\mathcal{Q}_h$ by adding the following constraint:

- For each job $v_j \in \mathcal{J}_1$ and for each basic interval b in P_h, only the *primary* time unit of b can be assigned to v_j, where the *primary* time unit of b, is u_i if b consists of three time units $u_{i-1}u_iu_{i+1}$, is u_1 if b consists of the first two time units u_1u_2 of $\mathcal{I}$, is u_I if b consists of the last two time units $u_{I-1}u_I$ of $\mathcal{I}$, is b itself if b consists of one time unit only.

Consider an optimal (unconstrained) scheduling M^*. For each job $v_j \in \mathcal{J}_2$, if M^* assigns v_j to two time units u_iu_{i+1}, then this assignment of v_j is also valid in exactly two problems among $\mathcal{Q}'_0$, $\mathcal{Q}'_1$, and $\mathcal{Q}'_2$, because there are exactly two indices $h \in \{0,1,2\}$ such that some basic interval in P_h contains both time units u_iu_{i+1}. Similarly, for each job $v_j \in \mathcal{J}_1$, if M^* assigns v_j to one time unit u_i, then this assignment of v_j is also valid in at least one problem among $\mathcal{Q}'_0$, $\mathcal{Q}'_1$, and $\mathcal{Q}'_2$, because there is at least one index $h \in \{0,1,2\}$ such that u_i is the primary time unit of some basic interval in P_h. Thus, by inheriting from the optimal scheduling M^*, the three problems $\mathcal{Q}'_h$ have more-constrained schedulings M^*_h such that M^*_h is a sub-scheduling of M^* and the three schedulings M^*_h altogether achieve at least a total profit of $w_1(M^*)+2w_2(M^*)$. Hence, the best more-constrained scheduling among M^*_1, M^*_2, and M^*_3 achieves at least a total profit of $\frac{1}{3}w_1(M^*)+\frac{2}{3}w_2(M^*)$. Indeed, we can prove the following better bound which is needed in later sections:

> The best more-constrained scheduling among M^*_1, M^*_2, and M^*_3 achieves a total profit of at least $\frac{1}{3}w_1(M^*)+\frac{2}{3}w_2(M^*)+\frac{1}{3}(p_1+p_I)$, where $p_1 = 0$ (respectively, $p_I = 0$) if M^* assigns no job in $\mathcal{J}_1$ to u_1 (respectively, u_I), while p_1 (respectively, p_I) equals the weight of the edge of M^* incident to u_1 (respectively, u_I) otherwise.

To see why we have this better bound, first note that there are exactly two indices $h \in \{0,1,2\}$ such that u_1 is the primary time unit of a basic interval in P_h. Similarly, there are exactly two indices $h \in \{0,1,2\}$ such that u_I is the primary time unit of a basic interval in P_h. By these two facts, the better bound follows.

As it should be expected, the constrained scheduling problems $\mathcal{Q}_h$ may often lead to better experimental results than the more-constrained scheduling problems $\mathcal{Q}'_h$. However, as for general theoretical results, we don't know if there is a difference between the two types of problems. Moreover, $\mathcal{Q}'_h$ can be solved more efficiently than $\mathcal{Q}_h$. Hence, for simplicity, in the following exposition we will consider only the more-constrained scheduling problems $\mathcal{Q}'_h$.

It is not hard to see that each more-constrained scheduling problem $\mathcal{Q}'_h$ requires the computation of a maximum-weight (unconstrained) matching in a suitably constructed bipartite graph G_h, and hence can be solved in polynomial time.

Lemma 2.1 *A constrained matching Z_1 in G can be found in $O(I(n_1+n_2)\sqrt{I+n_1+n_2})$ time, whose weight is at least $\frac{1}{3}w_1(M^*)+\frac{2}{3}w_2(M^*)+\frac{1}{3}(p_1+p_I)$, where $p_1 = 0$ (respectively, $p_I = 0$) if u_1 (respectively, u_I) is not matched to a vertex of $\mathcal{J}_1$ by M^*, while p_1 (respectively, p_I) equals the weight of the edge of M^* incident to u_1 (respectively, u_I) otherwise.*

Corollary 2.2 *If $w_1(M^*) \le (\frac{1}{2}-3\epsilon)w(M^*)$, then $w(Z_1) \ge (\frac{1}{2}+\epsilon)w(M^*)$.*

2.2 Preparing for the other three algorithms

Before running the other three algorithms, we need to compute a maximum-weight unconstrained matching M^*_{un} of G. The unconstrained matching M^*_{un} will be an additional input to the other three algorithms. Therefore, before proceeding to the details of the algorithms, fix a maximum-weight unconstrained matching M^*_{un} of G.

The algorithms in Sections 2.3 through 2.5 will use M^*_{un} in a sophisticated way. But first, we use M^*_{un} to define several subsets of U as follows.

- $U_0 = \{u_i \in U \mid u_i$ is not matched by $M^*_{\rm un}\}$.
- $U_1 = \{u_i \in U \mid u_i$ is matched to a $v_j \in \mathcal{J}_1$ by $M^*_{\rm un}\}$.
- $U_{2,1} = \{u_i \in U \mid u_i$ is matched to a $v_j \in \mathcal{J}_2$ by $M^*_{\rm un}\}$.
- $U_{2,2} = \{u_i \in U \mid u_i$ is matched to a $v_j \in \mathcal{J}'_2$ by $M^*_{\rm un}\}$.
- $W = \{u_i \in U_1 \mid u_{i-1} \in U_{2,1}$ and $u_{i+1} \in U_{2,2}\}$.
- $W_L = \{u_i \in U \mid u_{i+1} \in W\}$ and $W_R = \{u_i \in U \mid u_{i-1} \in W\}$.

In general, whenever $u_i \in W$, we have $u_{i-1} \in W_L$ and $u_{i+1} \in W_R$. Moreover, since $W \subseteq U_1$, no two sets among W, W_L and W_R can intersect.

A common idea behind the forthcoming algorithms is to divide the weights $w_1(M^*)$ and $w_2(M^*)$ into smaller parts, based on the aforementioned subsets of U. Define the smaller parts as follows.

- β_L is the total weight of all edges $(u_i, v_j) \in M^*$ such that $u_i \in W_L$ and $v_j \in \mathcal{J}_1$.
- β is the total weight of all edges $(u_i, v_j) \in M^*$ such that $u_i \in W$ and $v_j \in \mathcal{J}_1$.
- β_R is the total weight of all edges $(u_i, v_j) \in M^*$ such that $u_i \in W_R$ and $v_j \in \mathcal{J}_1$.
- $\bar{\beta} = w_1(M^*) - \beta_L - \beta - \beta_R$.
- α_0 is the total weight of all edges $(u_i, v_j) \in M^*$ such that either $v_j \in \mathcal{J}_2$ and $\{u_i, u_{i+1}\} \cap W = \emptyset$, or $v_j \in \mathcal{J}'_2$ and $\{u_{i-1}, u_i\} \cap W = \emptyset$.
- α_1 is the total weight of all edges $(u_i, v_j) \in M^*$ such that either $v_j \in \mathcal{J}_2$ and $\{u_i, u_{i+1}\} \subseteq W_L \cup W \cup W_R$, or $v_j \in \mathcal{J}'_2$ and $\{u_{i-1}, u_i\} \subseteq W_L \cup W \cup W_R$.

Lemma 2.3 *$\alpha_0 + \alpha_1 = w_2(M^*)$ and $\beta_L + \beta + \beta_R + \bar{\beta} = w_1(M^*)$.*

Now, we are ready to explain how the four algorithms are related. The algorithm in Section 2.3, called Algorithm 2, will output a constrained matching of weight at least $\frac{1}{3}\bar{\beta} + \frac{2}{3}\alpha_0 + \beta + \frac{2}{3}(\beta_L + \beta_R)$. The algorithm in Section 2.4, called Algorithm 3, will output a constrained matching of weight at least $\beta + \bar{\beta} + \alpha_1$. Thus, if $\beta \geq (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$, then Algorithm 2 or 3 will output a constrained matching of weight at least $(\frac{1}{2} + \epsilon)w(M^*)$ (see Corollary 2.6 below). On the other hand, if $\beta < (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$, then Algorithm 1 or 4 will output a constrained matching of weight at least $(\frac{1}{2} + \epsilon)w(M^*)$ (see Section 2.6).

2.3 Algorithm 2

We first explain the idea behind the algorithm. The removal of the vertices in W leaves $|W| + 1$ blocks of U each of which consists of consecutive vertices of U. For each block b, we use the idea of Algorithm 1 to construct three graphs $G_{b,0}, G_{b,1}, G_{b,2}$. For each $h \in \{0, 1, 2\}$, we consider the graph $\cup_b G_{b,h}$ where b ranges over all blocks, and obtain a new graph G'_h from $\cup_b G_{b,h}$ by adding the vertices of W and the edges $\{u_i, v_j\}$ of G such that $u_i \in W$ and $v_j \in \mathcal{J}_1$. We then compute a maximum-weight (unconstrained) matching in each G'_h, and further convert it to a constrained matching $\bar{M}'_h$ of G as in Algorithm 1. The output of Algorithm 2 is the heaviest matching among $\bar{M}'_0, \bar{M}'_1, \bar{M}'_2$.

Lemma 2.4 *A constrained matching Z_2 in G can be found in $O(I(n_1 + n_2)\sqrt{I + n_1 + n_2})$ time, whose weight is at least $\frac{1}{3}\bar{\beta} + \frac{2}{3}\alpha_0 + \beta + \frac{2}{3}(\beta_L + \beta_R)$.*

2.4 Algorithm 3

We first explain the idea behind Algorithm 3. Suppose that we partition the time interval $\mathcal{I}$ into shorter intervals in such a way that each shorter interval consists of either one time unit or three time units $u_{i-1}u_iu_{i+1}$ where $u_i \in W$. There is only one such partition of $\mathcal{I}$. Further suppose that we want to execute at most one job in each of the shorter intervals, while maximizing the total profit of the executed jobs. This problem can be solved in polynomial time by computing a maximum-weight (unconstrained) matching in a suitably constructed bipartite graph. We can prove that this matching results in a scheduling by which the executed jobs achieve at least a total profit of $\beta + \bar{\beta} + \alpha_1$.

Lemma 2.5 *A constrained matching Z_3 in G can be found in $O(I(n_1 + n_2)\sqrt{I + n_1 + n_2})$ time, whose weight is at least $\beta + \bar{\beta} + \alpha_1$.*

Corollary 2.6 *If $\beta \geq (\frac{1}{6} + \frac{5}{3}\epsilon)w(M^*)$, then $\max\{w(Z_2), w(Z_3)\} \geq (\frac{1}{2} + \epsilon)w(M^*)$.*

2.5 Algorithm 4

The idea behind Algorithm 4 is to convert $M^*_{\rm un}$ to a constrained matching of G. To convert $M^*_{\rm un}$, we partition $U_1 \cup U_{2,1}$ (respectively, $U_1 \cup U_{2,2}$) into two subsets none of which contains two vertices u_i and u_{i+1} such that $u_i \in U_{2,1}$ (respectively, $u_{i+1} \in U_{2,2}$). The set of edges of $M^*_{\rm un}$ incident to the vertices of each such subset can be extended to a constrained matching of G. In this way, we obtain four constrained matchings of G. Algorithm 4 outputs the heaviest one among the four matchings. We can prove that the weight of the output matching is at least $w(M^*_{\rm un})/2$.

We next proceed to the details of Algorithm 4. Algorithm 4 computes a constrained matching in G as

1. Starting at u_1, divide U into segments each of which is in the following form:

$$u_{i-\ell}u_{i-\ell+1}\cdots u_{i-1}u_iu_{i+1}\cdots u_{i+r-1}u_{i+r},$$

where $u_j \in U_{2,1}$ for all $i-\ell \le j \le i-1$, $u_j \in U_{2,2}$ for all $i+1 \le j \le i+r$, $u_{i-\ell-1} \notin U_{2,1}$, $u_{i+r+1} \notin U_{2,2}$, and u_i has no restriction. Note that ℓ and/or r may be equal to zero. We call u_i the *center* of the segment. For each segment s, let $c(s)$ denote the integer i such that u_i is the center of s; let $\ell(s)$ denote the number of vertices in s that precede $u_{c(s)}$; let $r(s)$ denote the number of vertices in s that succeed $u_{c(s)}$.

2. For each segment s, compute two integers x_s and y_s as follows:
   - If $u_{c(s)} \in U_0$, then $x_s = c(s)-1$ and $y_s = c(s)+1$.
   - If $u_{c(s)} \in U_1$, then $x_s = y_s = c(s)$.
   - If $u_{c(s)} \in U_{2,1}$, then $x_s = c(s)$ and $y_s = c(s)+1$.
   - If $u_{c(s)} \in U_{2,2}$, then $x_s = c(s)-1$ and $y_s = c(s)$.

3. Let $U^e_{2,1} = \bigcup_s\{u_i \mid (x_s-i) \bmod 2 = 0, c(s)-\ell(s) \le i \le x_s\}$,
   $U^o_{2,1} = \bigcup_s\{u_i \mid (x_s-i) \bmod 2 = 1, c(s)-\ell(s) \le i \le x_s\}$,
   $U^e_{2,2} = \bigcup_s\{u_i \mid (i-y_s) \bmod 2 = 0, y_s \le i \le c(s)+r(s)\}$,
   $U^o_{2,2} = \bigcup_s\{u_i \mid (i-y_s) \bmod 2 = 1, y_s \le i \le c(s)+r(s)\}$,
   where s runs over all segments.

4. Let $M^e_{2,1} = \{(u_i,v_j) \in M^*_{\rm un} \mid u_i \in U^e_{2,1}\} \cup \{(u_{i+1},v_{j+1}) \mid u_i \in U^e_{2,1} \cap U_{2,1}$ and $\{u_i,v_j\} \in M^*_{\rm un}\}$,
   $M^o_{2,1} = \{(u_i,v_j) \in M^*_{\rm un} \mid u_i \in U^o_{2,1}\} \cup \{(u_{i+1},v_{j+1}) \mid u_i \in U^o_{2,1} \cap U_{2,1}$ and $\{u_i,v_j\} \in M^*_{\rm un}\}$,
   $M^e_{2,2} = \{(u_i,v_j) \in M^*_{\rm un} \mid u_i \in U^e_{2,2}\} \cup \{(u_{i-1},v_{j-1}) \mid u_i \in U^e_{2,2} \cap U_{2,2}$ and $\{u_i,v_j\} \in M^*_{\rm un}\}$,
   $M^o_{2,2} = \{(u_i,v_j) \in M^*_{\rm un} \mid u_i \in U^o_{2,2}\} \cup \{(u_{i-1},v_{j-1}) \mid u_i \in U^o_{2,2} \cap U_{2,2}$ and $\{u_i,v_j\} \in M^*_{\rm un}\}$.
   Note that for each edge $(u_i,v_j) \in M^o_{2,1} \cup M^o_{2,2}$, we have $v_j \notin \mathcal{J}_1$. Indeed, $U^o_{2,1} \subseteq U_{2,1}$ and $U^o_{2,2} \subseteq U_{2,2}$.]

5. For the set $\bar{U}^o_{2,1}$ of vertices of U that are not matched by $M^o_{2,1}$, compute a maximum-weight matching $N^o_{2,1}$ between the vertices in $\bar{U}^o_{2,1}$ and the vertices in $\mathcal{J}_1$.

6. For the set $\bar{U}^o_{2,2}$ of vertices of U that are not matched by $M^o_{2,2}$, compute a maximum-weight matching $N^o_{2,2}$ between the vertices in $\bar{U}^o_{2,2}$ and the vertices in $\mathcal{J}_1$.

7. Output the maximum-weight matching Z_4 among $M^e_{2,1}$, $M^o_{2,1} \cup N^o_{2,1}$, $M^e_{2,2}$, $M^o_{2,2} \cup N^o_{2,2}$.

Lemma 2.7 *$M^e_{2,1}$, $M^o_{2,1} \cup N^o_{2,1}$, $M^e_{2,2}$ and $M^o_{2,2} \cup N^o_{2,2}$ are constrained matchings in G.*

Lemma 2.8 $w(M^e_{2,1}) + w(M^o_{2,1}) + w(M^e_{2,2}) + w(M^o_{2,2}) \ge 2w(M^*_{\rm un})$.

Lemma 2.9 $(U-\bar{U}^o_{2,1}) \cap (U-\bar{U}^o_{2,2}) \subseteq W$.

2.6 Performance of the algorithm when β is small

For a contradiction, assume the following:

Assumption 2.10 *$\beta < (\frac{1}{6}+\frac{5}{3})w(M^*)$ and $\max\{w(Z_1),w(Z_4)\} < (\frac{1}{2}+\epsilon)w(M^*)$.*

We want to derive a contradiction under this assumption. First, we derive three inequalities from this assumption and the lemmas in Section 2.5.

Lemma 2.11 $w(M^o_{2,1}) + w(M^o_{2,2}) \ge (1-2\epsilon)w(M^*)$.

Lemma 2.12 $w(N^o_{2,1}) + w(N^o_{2,2}) < 4\epsilon w(M^*)$.

Lemma 2.13 $\beta > w_1(M^*) - 4\epsilon w(M^*)$.

Now, we are ready to get a contradiction. By Corollary 2.2 and Assumption 2.10, $w_1(M^*) > (\frac{1}{2}-3\epsilon)w(M^*)$. Thus, by Lemma 2.13, $\beta > (\frac{1}{2}-7\epsilon)w(M^*)$. On the other hand, by Assumption 2.10, $\beta < (\frac{1}{6}+\frac{5}{3})w(M^*)$. Hence, $\frac{1}{2}-7\epsilon < \frac{1}{6}+\frac{5}{3}\epsilon$, contradicting our choice that $\epsilon = \frac{1}{26}$. Therefore,

Theorem 2.14 *A constrained matching Z in G with $w(Z) \ge \frac{13}{7}w(M^*)$ can be found in $O(I(n_1+n_2)\sqrt{I+n_1+n_2})$*

3 2-ISP with a special profit function

In this section, we consider *proportional 2-ISP*, where the profit of executing a job at each specific time interval is either 0 or proportional to the length of the job. A $\frac{5}{3}$-approximation algorithm was recently presented in [4] for proportional 2-ISP. Here, we present a $(1.5+\epsilon)$-approximation algorithm for it for any $\epsilon > 0$. We note in passing that a simple modification of this algorithm leads to a $(1.5+\epsilon)$-approximation algorithm for unweighted 2-ISP.

Let U, $\mathcal{J}_1$, and $\mathcal{J}_2$ be as in Section 2. Let E be the set of those $(u_i, v_j) \in U \times \mathcal{J}_1$ such that the profit of executing job v_j in time unit u_i is positive. Let F be the set of those $(u_i, u_{i+1}, v_j) \in U \times U \times \mathcal{J}_2$ such that the profit of executing job v_j in time units u_i and u_{i+1} is positive.

Consider the hypergraph $H = (U \cup \mathcal{J}_1 \cup \mathcal{J}_2, E \cup F)$ on vertex set $U \cup \mathcal{J}_1 \cup \mathcal{J}_2$ and on edge set $E \cup F$. Obviously, proportional 2-ISP becomes the problem of finding a matching $E' \cup F'$ in H with $E' \subseteq E$ and $F' \subseteq F$ such that $|E'| + 2|F'|$ is maximized over all matchings in H. Our idea is to reduce this problem to the problem of finding a maximum cardinality matching in a 3-uniform hypergraph (*i.e.* each hyperedge consists of exactly three vertices). Since the latter problem admits a $(1.5+\epsilon)$-approximation algorithm [6] and our reduction is approximation preserving, it follows that proportional 2-ISP admits a $(1.5+\epsilon)$-approximation algorithm.

Theorem 3.1 *For every $\epsilon > 0$, there is a polynomial-time $(1.5+\epsilon)$-approximation algorithm for proportional 2-ISP.*

4 A new heuristic for protein NMR peak assignment

As mentioned in Section 1, the $\frac{13}{7}$-approximation algorithm for 2-ISP can be easily incorporated into a heuristic framework for protein NMR peak assignment introduced in [7]. The heuristic first tries to assign "long" segments of three or more spin systems that are under the consecutivity constraint to segments of the host protein sequence, using a simple greedy strategy, and then solves an instance of 2-ISP formed by the remaining unassigned spin systems and amino acids. The first step of the framework is also called *greedy filtering* and may potentially help improve the accuracy of the heuristic significantly in practice because we are often able to assign long segments of spin systems with high confidence. We have tested the new heuristic based on the $\frac{13}{7}$-approximation algorithm for 2-ISP and compared the results with two of the best approximation and heuristic algorithms in [3, 4, 7], namely the 2-approximation algorithm for the interval scheduling problem [3, 4] and the branch-and-bound algorithm (augmented with greedy filtering) [7]. The test data consists of 70 (pseudo) real instances of NMR peak assignment derived from 14 proteins, each with 5 (density) levels of consecutivity constraints, as shown in Table 1. Each protein is represented as an entry in the BioMagResBank database [9], *e.g.* bmr4027, and the consecutivity level is represented by the underscore symbol following the BioMagResBank entry. For example, _5 means that the number of pairs of consecutive spin systems in the input is 50% of the total number of spin systems. Hence, the higher the consecutivity level index, the more the constraint. The program of the new heuristic is available to the public upon request to the authors.

References

[1] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48:1069–1090, 2001.

[2] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *Proceedings of STOC'99.*

[3] Z.-Z. Chen, T. Jiang, G. Lin, J. Wen, D. Xu, J. Xu, and Y. Xu. Approximation algorithms for NMR spectral peak assignment. To appear in *Theoretical Computer Science*, 2002.

[4] Z.-Z. Chen, T. Jiang, G. Lin, J. Wen, D. Xu, and Y. Xu. Improved approximation algorithms for NMR spectral peak assignment. *Proceedings of WABI'2002.*

[5] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Proceedings of FOCS'2001.*

[6] C.A.J. Hurkens and A. Schrijver. On the size of systems of sets of every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68-72, 1989.

[7] G. Lin, D. Xu, Z.-Z. Chen, T. Jiang, J. Wen, and Y. Xu. An efficient branch-and-bound algorithm for the assignment of protein backbone NMR peaks. *Proceedings of CSB'2002.*

[8] National Institute of General Medical Sciences. Pilot projects for the protein structure initiative (structural genomics). *http://www.nih.gov/grants/guide/rfa-files/RFA-GM-99-009.html*, June:RFA GM–99–009, 1999.

[9] University of Wisconsin. *BioMagResBank. http://www.bmrb.wisc.edu.* University of Wisconsin, Madison, Wisconsin, 2001.

[10] Y. Xu, D. Xu, D. Kim, V. Olman, J. Razumovskaya, and T. Jiang. Automated assignment of backbone NMR peaks using constrained bipartite matching. *IEEE Computing in Science & Engineering*, 4:50–62, 2002.